SYSTEMS ENGINEERING
Research Center

Assessing the Impact of Development Disruptions and Dependencies in Analysis of Alternatives of System-of-Systems

A013 – FInal Technical Report Phase I - SERC-2013-TR-035-2

March 04, 2013

Principal Investigator: Dr. Daniel A. DeLaurentis, Purdue University

Co-Principal Investigator: Dr. Karen Marais, Purdue University

Team Members

Navindran Davendralingam, Senior Research Associate, Purdue University

Seung Yeob Han, PhD Student, Purdue University

Payuna Uday, PhD Student, Purdue University

Zhemei Fang, PhD Student, Purdue University

Cesare Guariniello, PhD Student, Purdue University

Ankur Mour, Masters Student, Purdue University

Contract Number: H98230-08-D-0171

WHS TO 029; RT 44-b

Report No. SERC-2013-TR-035-2
March 04, 2013
UNCLASSIFIED

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
04 MAR 2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Assessing the Impact of Development Disruptions and Dependencies in Analysis of Alternatives of System-of-Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Purdue University,West Lafayette,IN,47907**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The development of emergent capabilities for a System-of-Systems (SoS) is greatly challenging due to the complex interdependencies between its constituent systems, that often exhibit managerial and operational independence. Current guidelines set forth by the Department of Defense SoS System Engineering guide presents SoS SE as a set of seven core elements which are connected to the 16 technical and management processes in the Defense Acquisition Guidebook (DAG). The DAG and subsequent frameworks (such as the DoD SoSE guide and wave model), are meant as base guidelines in addressing key issues associated with the management and evolution of an SoS gamut. Prior funded work, under RT-36, explored analytical methods to quantify the impact of system interdependencies in the context of SoS capability development and use this to guide system engineering activity for SoS. A variety of approaches were investigated to provide a means to conduct analysis of alternatives while navigating the decision space that simultaneously considers the potential positive impacts of interdependencies (e.g., SoS capability) as well as the negative impacts (e.g. consequences of disruption in development). The work is being continued under this RT-44 initiative, with the end goal of providing an analytical workbench complemented by intuitive and instructive applications on a naval scenario case study. This report details Phase I progress of a (Phase I & II) effort under RT-44 towards the development of a computational analytic workbench. Progress for each of the constituent methods, that comprise the range of tools in the workbench, is presented. Additionally improvements to the Littoral Combat Ship (LCS) inspired Naval Warfare Scenario (NWS) simulation, that serves as our representative SoS problem, is presented as well.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**46**	

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

ABSTRACT

The development of emergent capabilities for a System-of-Systems (SoS) is greatly challenging due to the complex interdependencies between its constituent systems, that often exhibit managerial and operational independence. Current guidelines set forth by the Department of Defense SoS System Engineering guide presents SoS SE as a set of seven core elements which are connected to the 16 technical and management processes in the Defense Acquisition Guidebook (DAG). The DAG and subsequent frameworks (such as the DoD SoSE guide and wave model), are meant as base guidelines in addressing key issues associated with the management and evolution of an SoS gamut.

Prior funded work, under RT-36, explored analytical methods to quantify the impact of system interdependencies in the context of SoS capability development and use this to guide system engineering activity for SoS. A variety of approaches were investigated to provide a means to conduct analysis of alternatives while navigating the decision space that simultaneously considers the potential positive impacts of interdependencies (e.g., SoS capability) as well as the negative impacts (e.g. consequences of disruption in development). The work is being continued under this RT-44 initiative, with the end goal of providing an analytical workbench, complemented by intuitive and instructive applications on a naval scenario case study.

This report details Phase I progress of a (Phase I & II) effort under RT-44 towards the development of a computational *analytic workbench*. Progress for each of the constituent methods, that comprise the range of tools in the workbench, is presented. Additionally, improvements to the Littoral Combat Ship (LCS) inspired Naval Warfare Scenario (NWS) simulation, that serves as our representative SoS problem, is presented as well.

This Page Intentionally Left Blank

TABLE OF CONTENTS

FIGURES AND TABLES

1 SUMMARY

The need for methods to balance risk and capability tradeoffs when assessing alternatives in decision making for SoS architectures has been a primary motivation for the research documented in this report. Trades between overarching capability and risk are essential decisions that must be addressed for SoS capability planning. Existing tools for such trades, where they exist, can be ineffective and non-intuitive when size and/or interdependency complexity is high. These features create a tradeoff space between development risk and capability potential of a system. The methods adapted in our Analytic Workbench support artifacts of the wave model in evolving a SoS architectures have been refined to the following:

- Robustness using Stand-In Redundancy
  The method employed here employs robust strategies in evaluating and constructing SoS networks of assets that have the ability to mitigate

- Interdependency and Resilience Analysis using Bayesian Networks
  This method utilizes statistical/distribution information from data driven SoS networks to address issues of network capability resilience and analysis of interdependencies in developmental networks.

- Functional (Developmental) Dependency Network Analysis (F(D)DNA)
  This method extends the Markov network approach to analysis of both functional and developmental dependencies between constituent systems in a SoS architecture.

- Architectural development using Colored Petri Nets (CPN)
  This method employs a discrete event simulation technique to analyze event based architectures; discrete event networks are modeled as a combination of tokens, transitions and rule sets.

- SoS Architecture Decision Analysis using Robust Portfolio Optimization
  This method provides a decision support framework for the identification of 'portfolios' of interdependent systems that fulfill requirements and capabilities.

This report is organized as follows: Section (2) of the report is the introduction and covers the background motivation behind the research in supporting SoS architectural evolution from the perspective of a SoS SE practitioner. Section (3) details progress in Phase 1 objectives of advancing our analytic methods, as listed above, to further support wave model artifacts in

evolving SoS architectures. Subsections of Section (3) reflect progress made for each method individually. A part of section 3 is also devoted to development progress of a Littoral Combat Ship (LCS) inspired Naval Warfare Scenario (NWS); this entails the extension of an agent based simulation scenario that serves as a synthetic case study for application of the workbench tools. The modeling also includes a computational mechanism design approach to managing sensor resource management as a further improvement of fidelity of our NWS case and an application of policy control in agent based models.

2 INTRODUCTION

The Department of Defense (DoD) has released a SoS SE guidebook, which presents SoS SE as seven core elements that are mapped to the original 16 technical management processes within the Defense Acquisition Guidebook (DAG) (Defense, Defense Acquisition Guidebook, 2008). The guidebook serves as a concerted effort in understanding how SoS architectures work and address a holistic view of what frameworks are necessary in tackling SoS SE challenges. However, the guidance provided in the document is still in need of more comprehensive guidelines and complementary technical tools to enable effective SoS SE management and support.

2.1 PROBLEM STATEMENT

While the wave model establishes a concise framework on addressing SoS architectural evolutions, there is nevertheless a pressing need for adequate tools to support each stage of the decision epochs, and to navigate the complexities associated with decision making in such environments. Trades between capability and risk are essential during analysis of alternatives for SoS capability planning, namely in the evaluation of alternative architectures. Existing tools for such trades, where they exist, can be ineffective and non-intuitive when size and/or interdependency complexity is high. These features of interdependencies create a tradeoff space between development risks, cost and capability performance of a SoS, for which new tools/methods will need to be developed.

2.2 RESEARCH OBJECTIVES

The objective of our effort, initiated in RT-36 and continues in RT-44b, is to develop and test methods that quantify the impact of system interdependencies in the context of SoS capability development, in support of the wave model. The products of the research can guide system engineering and architecture design activities. A variety of methods are investigated to conduct analysis of alternatives while navigating the decision space that simultaneously considers the potential positive impacts of interdependencies (e.g., SoS capability) and negative impacts (e.g. consequences of disruption). This report details progress towards supporting RT-44b Phase 1 general objectives:

- *Analytic workbench* concept platform generation; consolidation of current researched methods
- Case study development; extension of the LCS inspired Naval Warfare Scenario (NWS).
- Prototype workbench application to NWS case study

Progress in the above Phase 1 objectives is supported through development of the individual methods that comprise the SoS workbench and the agent-based simulation of our NWS case study. We envision our analytic workbench as part of an iterative feedback process, in support of artifacts of the wave model, as depicted in the following Figure (1).

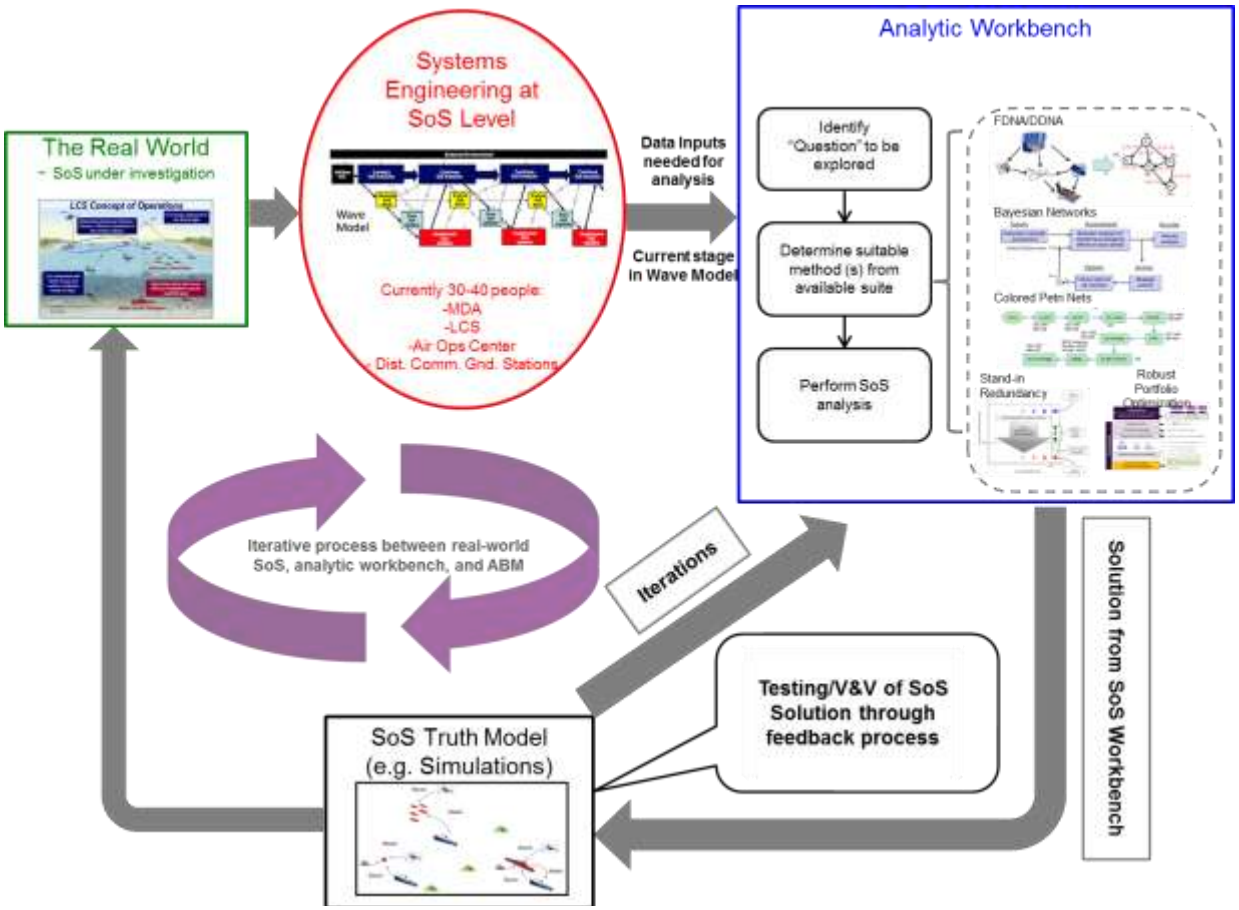

Figure 1: SoS Analytic Workbench concept of use

Figure (1) shows the concept of use of our analytic in support of evolving a SoS. The main idea is that data/information on the current state of systems in an SoS (following stage definitions within the wave model), can be brought into an analytic environment of the *Analytic Workbench*. The suite of tools of the workbench can be brought to bear in addressing

archetypal analysis on evolving SoS architectures;. Figure (2) below maps some of the main archetypical forms of analysis that can be mapped to tools in the current analytic workbench.

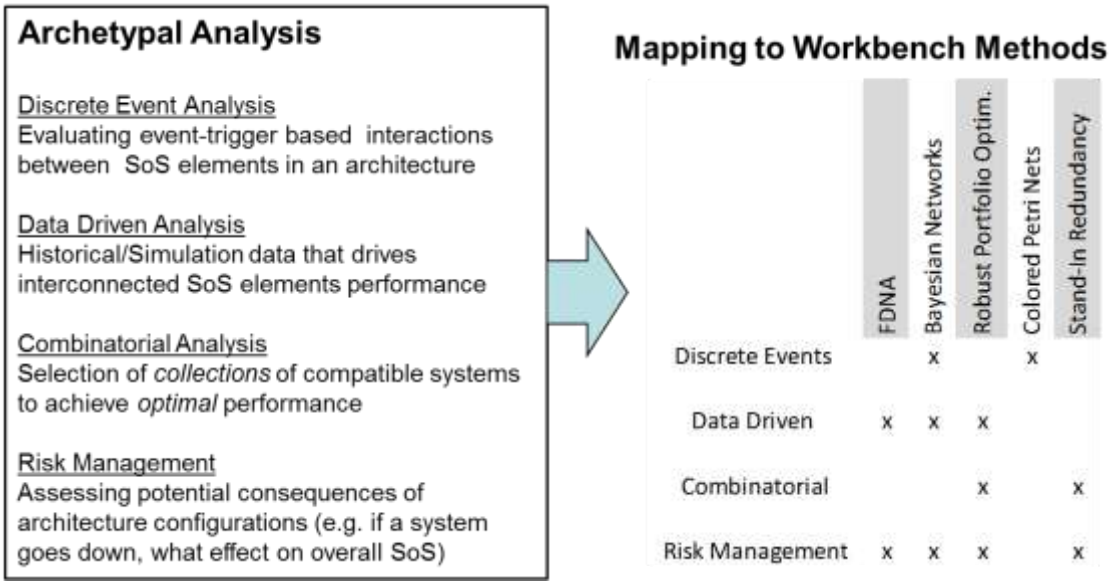

Figure 2: Archetypal mapping to methods

The iterative process as shown in Figure (1) , in addressing the archetypal analyses of Figure (2), is executed in concert with available 'truth models' (e.g. computational simulations, field testing) in providing preliminary verification of the next SoS evolution *solution* The *solution* in this case refers to suggested architectural changes (e.g. addition/removal of systems and/or links) towards fulfilling target SoS capabilities, while preserving acceptable risk (operational or developmental) and cost. Each method requires a set of prescribed inputs/data as listed in Table 1 below.

Table 1: Inputs for workbench methods

Methods	Required Input Data Elements
FDNA	COD, SOD, Inter-system connectivities
Bayesian Networks	Inter-system directional connectivities Probability distributions of system capabilities
Robust Portfolio Optimization	Capabilities, development risk, Compatibilities, System cost
Petri Nets	Discrete event rules, System capabilities, Connectivities
Stand-In Redundancy	System capabilities, development risk, Inter-system compatibilities

3 Analytic Methods: Phase 1 Progress

3.1 Interdependency modeling for System-of-Systems (SoS) using Bayesian Networks

Interdependencies between systems, while allowing SoS to achieve their objectives, also permit the propagation of failure modes throughout the SoS. For example, in a ballistic missile defense system, if integrated sensor system fails to detect a ballistic missile, then the interceptor system will also fail in achieving its objective. These properties of independence and interdependence make evaluating SoS designs challenging. We adopt a probabilistic approach to addressing issues related to SoS interdependencies, using a Bayesian network approach.

3.1.1 Overview of Previous work from RT-36

We developed a method based on Bayesian networks to evaluate the resilience of SoS design alternatives against failure during operations. When designing SoS, engineers and decision makers are interested in identifying patterns of performance in response to failures in component systems. A metric that quantifies the capacity of a system to withstand failures can aid in design and operations choices. *SoS resiliencies defined as the level of performance achieved relative to different levels of failure*. In order to achieve this objective, we addressed three questions:

How can we represent systems for resilience analysis?

We briefly reviewed the research on hierarchical representations and applied a hierarchical network representation to decompose SoS into three levels: capability, requirements, and systems.

How can we analyze system interdependencies?

We used a Bayesian Network model to analyze interdependencies between capability, requirements, and systems and to address uncertainty. This model captured the patterns of how individual system failures propagate to other dependent systems. These patterns allowed us not only to identify critical subsystems that are responsible for potential system failures, but also to analyze SoS architecture alternatives.

How can we measure system resilience?

We defined metrics for assessing resilience. Then, we identified several patterns of resilience for different SoS architectures.

After proposing ways of representing and evaluating SoS resilience using Bayesian networks and a resilience metric, we demonstrated our approach using a Naval Warfare Scenario (NWS) example, that is inspired by the Littoral Combat Ship (LCS) mode of operations. SoS resilience was determined by both the SoS architecture and the constituent system reliability. In the proposed example, we determined the two most critical systems using the conditional resilience metric. Adding a communications link between these two systems increased the resilience, resulting in higher expected performance and slower expected performance degradation as a result of system failure. Detailed results on the application are as published in the SERC RT-36 Final Report (Report Number: SERC-2012-TR-036).

3.1.2 Current Progress: Failure Probability of Constituent Systems in the NWS SoS

When data driven methods such as Bayesian Networks are used for an interdependency analysis, appropriate failure probabilities of constituent systems in the network of systems is important role in providing right results. Therefore it is important to choose a right model to estimate appropriate failure probabilities of constituent systems. These probabilities can be determined based on operational experience, which can for example be used to determine the mean-time-between-failure (MTBF). Alternatively, they can be estimated during design using techniques such as fault tree analysis or agent-based models. The Naval Warfare Scenario (NWS) SoS is a special case because they are routinely exposed to exogenous attacks and therefore their failure cannot be predicted by solely considering endogenous factors like ageing. Therefore a competing risk model is used to describe the failure probabilities of constituent systems in the NWS SoS (Bocchetti, Giorgio, Guida, & Pulcini, 2009). To illustrate the process, the probability of failing due to aging and hostile attack is considered.

The overall hazard rate for all the causes is:

$$\lambda(t) = \lim_{dt \to 0} \frac{p\{t \le T < t + dt \mid T \ge t\}}{dt} \quad (1)$$

where T is a continuous random variable describing the failure time .
The cause-specific hazard rate is the hazard of failing from the jth cause in the presence of the competing events from all the causes:

$$\lambda_j(t) = \lim_{dt \to 0} \frac{p\{t \le T < t + dt,\ J = j \mid T \ge t\}}{dt} \quad (2)$$

The cumulative cause-specific hazard rate is then defined by integrating the overall hazard rate:

$$\Lambda(t) = \int_0^t \lambda(u)du \tag{3}$$

If the competing risks are independent, the marginal survival function follows directly from the definition of the hazard rate:

$$S(t) = \exp\left(-\Lambda(t)\right) \tag{4}$$

By the law of total probability, the total hazard λ(t) and the overall survival function S(t) can be defined in terms of the cause-specific hazards:

$$\lambda(t) = \lim_{dt \to 0} \frac{p\{t \le T < t + dt \mid T \ge t\}}{dt} = \sum_{j=1}^{2} \lambda_j(t) \tag{5}$$

$$S(t) = p(T > t) = \exp\left(-\Lambda(t)\right) = \exp\left(-\sum_{j=1}^{2} \Lambda_j(t)\right) \tag{6}$$

where $-\Lambda_j(t)$ is the integrated or cumulative hazard for case j. The function S(t) is interpreted as the probability of surviving both causes of failure up to time t.
The cumulative incidence function of cause j is defined by:

$$F_j(t) = p(T \le t, J = j), \quad j = 1,2 \tag{7}$$

This function provides the probability of a system failure caused by a failure mode j in the presence of all the competing risks during the system's entire life. It can be represented in terms of the cause-specific hazard and the overall survival function:

$$F_j(t) = \int_0^t \lambda_j(u) S(u) du, \quad j = 1,2 \tag{8}$$

Finally, the overall cumulative distribution function of a system failure under the competing risk model at time t is:

$$F(t) = \sum_{j=1}^{2} F_j(t), \quad j = 1,2 \tag{9}$$

Systems may exhibit many different failure rates as a function of time, the bathtub curve here is used as it allows observing several different failure modes (Dhillon, 2006). Further, since the systems in an SoS are usually complex themselves, it is not unreasonable to expect bathtub behavior (Briand, Lowder, & Shirah, October 2006). The bathtub hazard rate curve can be divided into three regions: 'Infant mortality', 'Useful life period', and 'Wear-out period' as shown in **Error! Reference source not found.**. During the infant mortality period, the system hazard rate decreases failures due such as poor manufacturing methods, human error, poor quality control, and substandard materials and workmanship decrease over time. During the useful life period, the system hazard rate remains constant due to causes such as undetectable defects and natural failures and due to explainable causes. During the wear-out period, the

system hazard rate increases as parts fail due to causes such as wear caused by friction, aging, and corrosion.

The bathtub hazard rate curve can be represented using a combination of Weibull distributions. Then the survivor function and hazard rate are:

$$S_A(t) = \exp\left[-\left(\frac{t}{\eta_1}\right)^{\beta}\right] \tag{10}$$

$$\lambda_A(t) = \frac{\beta}{\eta_1}\left(\frac{t}{\eta_1}\right)^{\beta-1} \tag{11}$$

where $\eta_1 > 0$ is the scale parameter and $\beta > 0$ is the shape parameter of the Weibull distribution. β determines the failure rate. If $\beta < 1$, the hazard rate decreases with time, if β = 1 hazard rate is constant, and if $\beta > 1$the hazard rate with time. **Error! Reference source not found.** shows an example of bathtub and constant hazard rates using exponential and Weibull distributions.

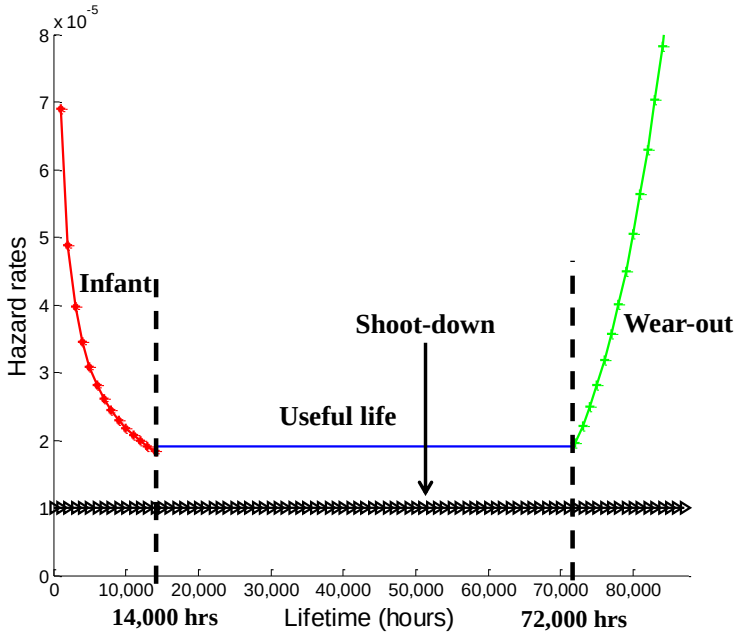

Figure 3: Constant and bathtub hazard rates using exponential and three different Weibull distributions

For the hostile attack failure mode, it is assumed that the failure rate is constant over time, therefore the survivor function is exponential:

$$S_S(t) = \exp(-\eta_0 t) \tag{12}$$

where $\lambda_S(t) = \eta_0$ is the constant hazard rate.

Under the assumption of independence between the two distributions, the reliability in the presence of competing risks is the product of the failure mode reliability functions (10) and (12):

$$S(t) = S_A(t)S_S(t) = \exp\left[-\eta_0 t - \left(\frac{t}{\eta_1}\right)^{\beta}\right] \quad (13)$$

Three cumulative functions are determined in the presence of competing risks by applying equations (8) and (9): the cumulative incident function of aging and hostile attack failure modes respectively and the overall cumulative distribution function of the system failure. **Error! Reference source not found.** depicts the estimate of the cumulative failure rate under the competing risk model together with the cumulative failure rates caused by two failure modes. Bathtub and constant hazard rates provided in **Error! Reference source not found.** are used to generate the cumulative failure probabilities. Under these hazard assumptions the system aging is the dominant failure mode over lifetime. It also shows that there are two places (around time 14,000 and 72,000 hours) on a cumulative failure distribution where noticeable slope changes exist due to severe transitions of hazard rates patterns between three regions.

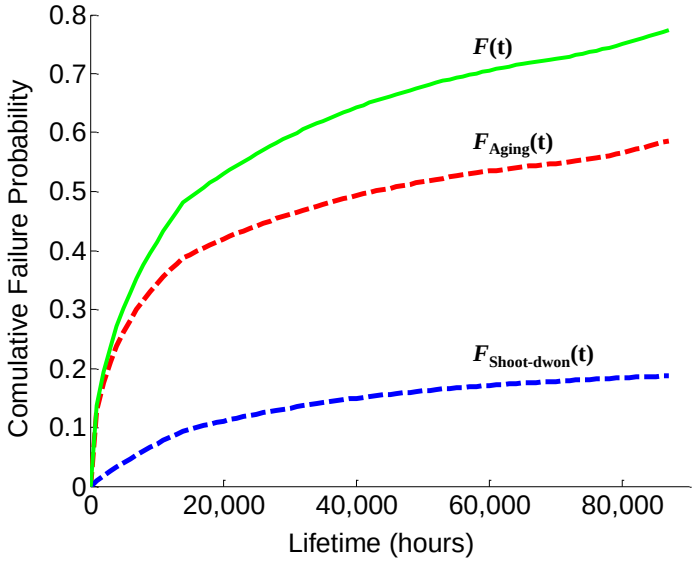

Figure 4: Estimate of the cumulative failure rates under the competing risk model and two failure modes

3.1.1 Conclusion and Future Work

For the case study of the NWS SoS, we defined a competing risk model that considers the characteristics of system failure modes (aging and hostile attack failures) to describe the failure probabilities of constituent systems. This model provided good estimations of appropriate failure probabilities of constituent systems. In future work we will integrate the competing risk model with interdependency modeling developed from RT-36 and identify patterns of

performance of the NWS SoS in response to failures in component systems using the proposed integrated model.

3.2 Naval Warfare System-of-Systems (SoS) Agent-based Model

An agent-based simulation model was developed to capture the performance of the Littoral Combat Ship (LCS) in a warfare scenario involving multiple mission threats. Purdue University has its very own MATLAB-based Agent-Based Modeling (ABM) called Discrete Agent Framework (DAF) which was used to generate the naval warfare model. Such full-scale littoral battle scenarios typically involve a Littoral Combat Ship (LCS) squadron with a mix of ships and accompanying helicopters, unmanned aerial and surface vehicles, and a host of antagonist units. The ABM itself was based on the models described in three masters' theses from US Naval Postgraduate School (Abbot 2008, Ozdemir 2009, Jacobson 2010). Military and industry experts reviewed the theses prior to final approval and verified their descriptions of the functionality and performance parameters of the individual components of the SoS.

3.2.1 Overview of Naval Warfare Model

The LCS was primarily designed to take advantage of the interchangeable modularized onboard packages and so each of these packages was modeled in the simulation scenario: Surface Warfare (SUW), Anti-Submarine Warfare (ASW), and Mine Counter-Measures (MCM). The protagonist forces were augmented by the presence of a host of supporting entities which included – MH-60R, Unmanned Aerial Vehicles (UAVs), Unmanned Surface Vehicles (USVs) and the Remote Mine-hunting Vehicles (RMVs). Each of the units has different capabilities and tackles different antagonist units, depending on the LCS mission package to which they are affiliated.

In order to accurately capture how LCS will perform in a stressing operational environment, a robust scenario containing three different types of antagonist units was developed – missile boats, submarines and mines. The missile boats and submarines exhibit different patrol, detection and engagement strategies while the mines simply detonate whenever an asset is within its proximity radius. Merchant traffic was also included to emulate realism in the scenarios and to add to the surface clutter, thereby making detection more difficult for both the protagonist and antagonist units.

The mission of the LCS fleet was to clear the waters of all threats, while incurring the minimum number of friendly casualties. The factors that played an important role in this

simulation are the number of enemy platforms, the number and type of LCS, the probability of detection for the friendly sensors, and the probability of kill for friendly weapons.

3.2.2 AGGREGATE RESULTS FROM MULTIPLE SIMULATION RUNS

Multiple simulation runs were carried out in order to analyze the battle scenario and the agent behavior. The figure below displays the starting and the emergent scenario that arises for a typical run case -

Table 2 Results from a Typical Simulation Run

Agent Terminated	Agent Location	Time Step	Terminator	Terminator Location	Weapon Used
Mine 5	(75,40)	38	MIW MH60	(78.85,32.18)	Clearance
Mine 4	(65,50)	200	MIW MH60	(68.7, 42.9)	Clearance
Mine 3	(55,60)	272	MIW MH60	(58.70,49.89)	Clearance
Mine 2	(35,65)	621	MIW MH60	(40.58,65.77)	Clearance
MBS 1	(17.8,111.18)	623	SUW LCS	(30.88,96.62)	NLOS
MBS 2	(17.96,110.09)	635	SUW LCS	(28.25,99.63)	NLOS
MBS 3	(18.65,112.08)	642	SUW LCS	(26.72,101.39)	NLOS
MBS 4	(19.2,112.56)	651	SUW LCS	(24.75,103.66)	NLOS
MBS 5	(18.10,108.88)	659	SUW LCS	(22.98,105.66)	NLOS
Mine 1	(15,75)	665	MIW MH60	(20.32,73.5)	Clearance
ASW LCS	(44.72,79.73)	763	Submarine	(44.69, 79.69)	Torpedo

The simulation results reveal expected outcome for missile boats and mines and non-intuitive results for the ASW LCS due to emergent properties. The SUW MH-60R helicopter detects missile boats early in the scenario due to its high velocity and detection radius and passes on the position information to the SUW LCS. Guided by the antagonist's position information, the SUW LCS was able to terminate each of the missile boats using its Non Line of Sight Launch System. The MIW LCS, on the other hand, detects the mines and passes on the position information to the MIW MH-60 and the other LCSs', so that they can avoid colliding with the mines. The MIW MH-60, guided by the antagonist's position information, terminates each of the mines using its Clearance Missiles.

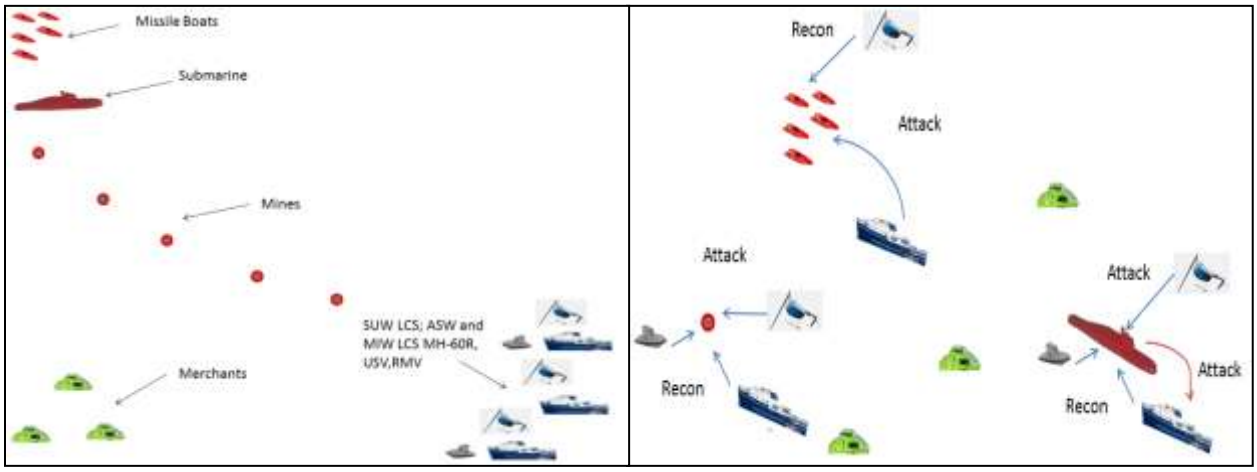

Figure 5 : Starting and Emergent Scenario

The termination of the ASW LCS at the hands of the submarine was an emergent and non-intuitive result. The ASW LCS does not have the means to terminate submarines and is restricted to maintain a standoff distance of 10 nautical miles from a detected submarine and to pass on the submarine's location to the accompanying MH-60R. Once the detected submarine is within the weapon range of the SUW MH-60R, it fires torpedo missiles on the submarine; however because of the limited number of torpedo missiles onboard the ASW MH-60R and the probability of termination associated with the torpedo missiles, the ASW MH-60R either missed its target or could only injure the submarine, before it ran out of ammunition. The submarine was then able to terminate the ASW LCS before the ASW MH-60R could reload its torpedo missiles. Once the ASW LCS was terminated, the ASW MH-60R could no longer reload and the submarine was able to escape.

3.2.3 Conclusion and Future Work

The Naval Warfare Simulation model provides a good insight into the workings of agent-based modeling and serves as an ideal exploratory model to glean the mission performance for a naval warfare scenario. The model lucidly captures the unfolding of emergent behavior and properties, because of the collective interaction among the system components and provides inroads in capturing the performance and the shortcomings of the LCS. The next phase would involve extending the Naval Warfare System-of-Systems Agent-based Model and extracting the different outputs, that needs to be plugged into the architecture evaluation methods as parameter values and inputs within the Analytic Workbench Framework.

3.3 COMPUTATIONAL MECHANISM DESIGN FOR SENSOR RESOURCE MANAGEMENT

Military platforms need to establish and maintain a common understanding of the tactical situation through the sharing and exchange of tactical data from sensors on each of the operationally independent platforms in the group, using a standardized radio net-work, commonly called a tactical data information link (TADIL). Sensor allocation takes place using "Reporting Responsibility" (R^2) rules which assign a track to the platform that has the best quality data for that track (Friedman 2006).The goal of the work is to provide improved scenario fidelity in our concept Naval Warfare Scenario; this is achieved through additional development of an auction mechanism to efficiently allocate a finite but selectable amount of bandwidth, above and beyond that already used in a conventional R^2 approach.

It is assumed that platforms are rational and therefore liable to deceptive behavior if it furthers their objectives. This might, in fact, be a reasonable assumption in TADILs that operate under multiple coalition flags. Each agent has private information about the quality of its data and can misreport this quality and degrade or otherwise decline to share its data. To deal with this problem we borrow the concept of Mechanism Design from the field of Microeconomics and Game Theory. Mechanism design involves designing mechanisms, and institutions that are mathematically proven to satisfy certain system-wide objectives under the assumption that individuals interacting through such institutions act in a self-interested manner and may hold private information that is relevant to a required decision.

3.3.1 OVERVIEW OF THE ABM

An agent-based simulation model was developed to capture the sensor resource management problem in a scenario involving multiple sensor platforms and tracks in DAF. The objective is to investigate the potential for using computational mechanisms in performance-critical, highly dynamic tactical data networks, to improve the quality of a combat group's common operating picture.

Sensor platforms are located at different positions on the simulation map. Each sensor has a detection radius wherein it can detect target tracks and a smaller classification radius wherein it can classify the detected targets as friendly, neutral or hostile. An auction is initiated as the beginning of the simulation to allocate the target tracks to each senor based on the information quality, and the auction is repeated after a finite number of time steps. Each sensor is responsible for tracking and reporting the position information and the group affiliation of the target tracks that are assigned to it.

In the first phase of the work, only the Reporting Responsibility approach has been modeled in the ABM. A GUI has been developed with the objective in mind that the end-user will be able to operate and visualize the simulation, without the need of interacting with DAF.

3.3.2 Results and Future Work

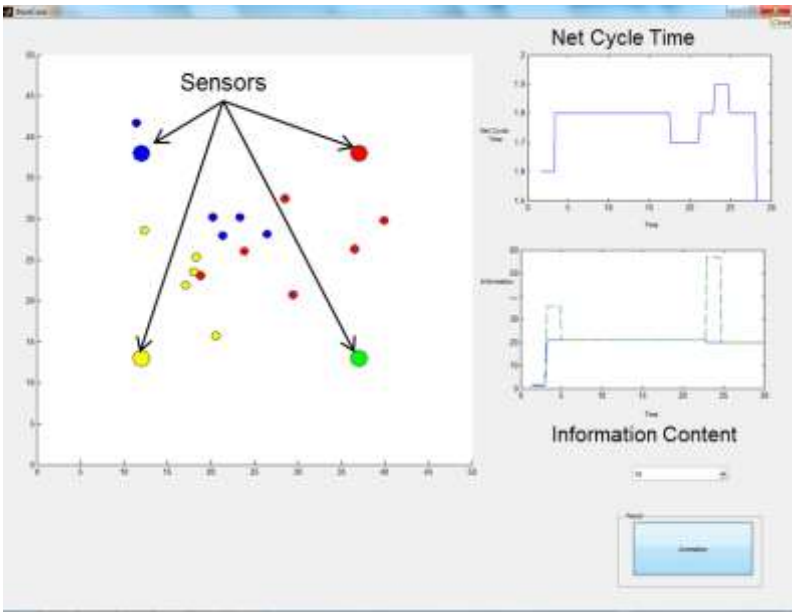

Figure 6: Sensor Allocation ABM GUI

Figure (6) above shows the ABM GUI in action. There are four sensors symmetrically located within the map, and the Red Sensor acts as the Grid Reference Unit (GRUs). GRU generally possess the highest quality track data and almost universally acquire R^2 for any tracks in its radius. The targets are color coded based on the sensor to which they have been assigned for tracking. The TADIL emulated in this application framework uses a round-robin approach to data transmission and the plot on the top indicates the total time for one round of transmission. The bottom plot shows the current and potential information content that can be exchanged at any given time.

The next phase of the work would involve using a mechanism design framework to develop an auction protocol, to recover the most valuable gain in the information for a given quantum of extra bandwidth.

3.4 Stand-In Redundancy

Resilience is the ability of a system or organization to react to and recover from disturbances with minimal effect on its dynamic stability (Hollnagel et al, 2010). While the resilience of system-of systems (SoSs) depends on the reliability of their constituent systems, traditional reliability approaches cannot adequately quantify their resilience. Given the heterogeneity and often wide geographic distribution of SoS constituent systems, inclusion of backup redundant systems for a SoS is usually impractical and costly. In this work, “stand-in redundancy” is presented as a way to quantitatively assess the impact of compensating for a loss of performance in one constituent

system by re-tasking the remaining systems. This concept and its resulting upstream effects on development costs and risks can be used by decision-makers to quantitatively assess the impact on resilience of different SoS architectures and their inherent ability to resist failures throughout the SoS lifecycle.

3.4.1 Overview of previous work from RT-36

Traditional systems engineering practices try to anticipate and resist disruptions through classical reliability methods, such as inclusion of redundancy at the component level and use of preventive maintenance at the system level. Reliability analysis techniques (Rausand and Hoyland, 2004), such as fault trees and event trees, are used to determine the level and types of redundancy to be included in the system design. Similar methods are used to develop maintenance plans to reduce the likelihood of failures at the system level. However, these approaches do not adequately satisfy the resilience needs of a SoS. Given the heterogeneity and, often wide geographic distribution, of the constituent systems, redundant systems in a SoS are impractical and costly. Additionally, high levels of interdependency between the systems imply increased risks of failures cascading throughout the SoS. These hurdles offer the opportunity to improve the resilience of the overarching system through unconventional means. In this work, a way to compensate for a loss of performance in one constituent system by re-tasking the remaining systems is studied. Specifically, as one entity, or node in a SoS, experiences degraded performance or a failure mode, other entities can alter their operations to compensate for this loss. This idea is termed “stand-in redundancy”, and it raises several interesting questions, such as: (1) given the failure of a specific system, what is the best configuration to compensate for the loss; (2) what level of performance can be recovered with the new configuration?; and (3) what is the upstream effect of stand-in redundancy on development costs and risks? These different kinds of resilience can be used to study various SoS architectures and evaluate their inherent ability to resist failures, thus in effect providing information for decision-makers to help identify “optimally” resilient SoS structures. Additionally, consideration of these resilience improvement techniques enables designers to better target risk resolution resources.

Error! Reference source not found. provides a graphical illustration of the value of stand-in redundancy in SoS operations using the basics metrics of performance and reliability that are essential characteristics of any system. The desirable region of operation is in the top right of the graph, that is, the high-performance, high-reliability portion (as indicated by the blue circle). After the initial deployment, the systems gradually degrade with time, and the SoS region of operation moves to the left of the graph. In some cases, this degradation can result in a simultaneous reduction in performance levels as well as reliability (not shown in figure). If a

system fails, the immediate loss in its functionality leads to a decrease in the overall performance level of the SoS (as shown by the red circle). Given the inherent characteristics of SoSs, a single system loss typically does not lead to a complete failure of the larger SoS and hence, the overall LoP does not fall to zero. However, we propose that incorporating a certain level of stand-in redundancy will allow the SoS to minimize this performance loss without relying on external agents to either maintain or replace the failed system. In the event of a system failure, by allowing multiple systems in the SoS to perform the same functions, the remaining systems in the SoS can be re-tasked to perform the lost functions, even if to a lesser degree of performance. This is represented by a sequence of green triangles, indicating that different levels of performance can be regained depending on the functional re-configurability of the SoS.

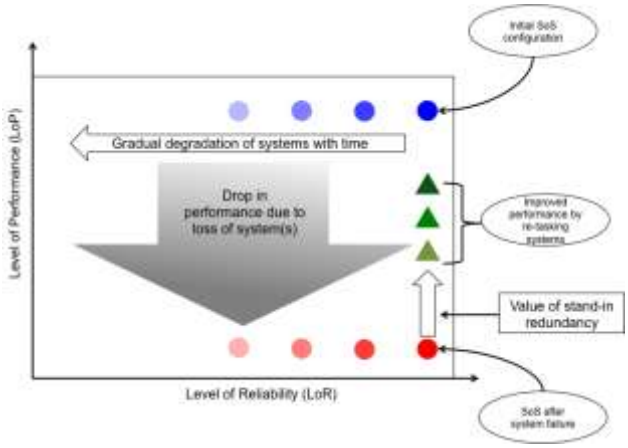

Figure 7: Notional Performance-Reliability graph showing impact of "stand-in redundancy" on SoS performance

3.4.2 Current Research Efforts (1): Method Validation using Naval Warfare ABM

The in-house Naval Warfare Scenario Agent-Based Model (ABM) consists of three different Littoral Combat Ship (LCS) packages: (1) surface, (2) anti-submarine, and (3) mine-warfare. These units are used to detect and eliminate antagonistic elements, such as missile boats, mines, and submarines. This ABM is used to study and validate the stand-in redundancy model developed in the first part of this work (as described in the previous section). First, the key inputs, outputs, and assumptions needed to integrate the model with the method, were identified. For example, the output of the ABM includes two values, namely, the time of enemy detection and the time of enemy elimination. These parameters were used to define the Level of Performance (LoP) of the overall SoS. Similarly, on the input side, to enable stand-in redundancy, factors such as detection range, detection probability, and the amount of

ammunition for each system could be varied. The variables of interest are shown in **Error! Reference source not found.**.

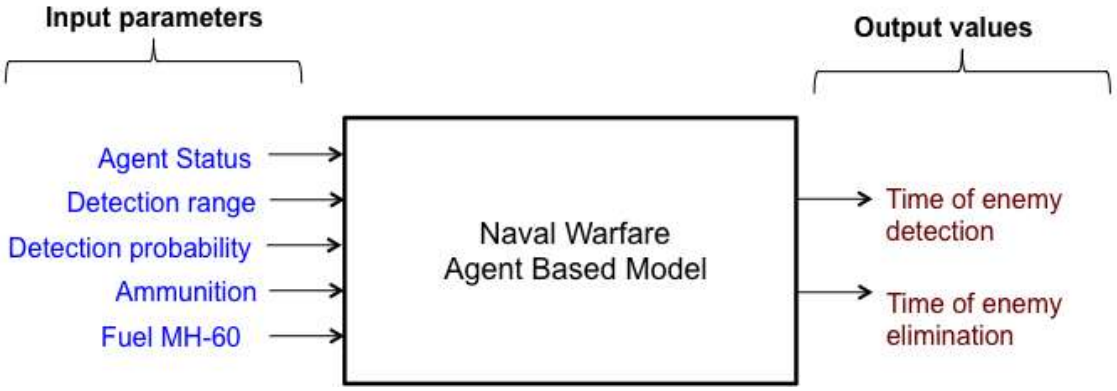

Figure 8 : Variables of interest

Given the working of and the variables included in the ABM, stand-in redundancy can be realized in this SoS by changing the nodes, links, or a combination of both (as shown in **Error! Reference source not found.**). Modification of the nodes implies changing the features available on the constituent system. For instance, the MH-60 helicopter can be made to carry a greater number of weapons, or the detection range of the unmanned surface vehicles could be increased indicating more sophisticated equipment on board. On the other hand, modifying the links involves allowing new communication links between systems that were not present in the original configuration. For example, in the event of a system failure, say the anti-submarine LCS, other systems associated with this LCS, such as the MH-60, can be networked with entities in the rest of the SoS.

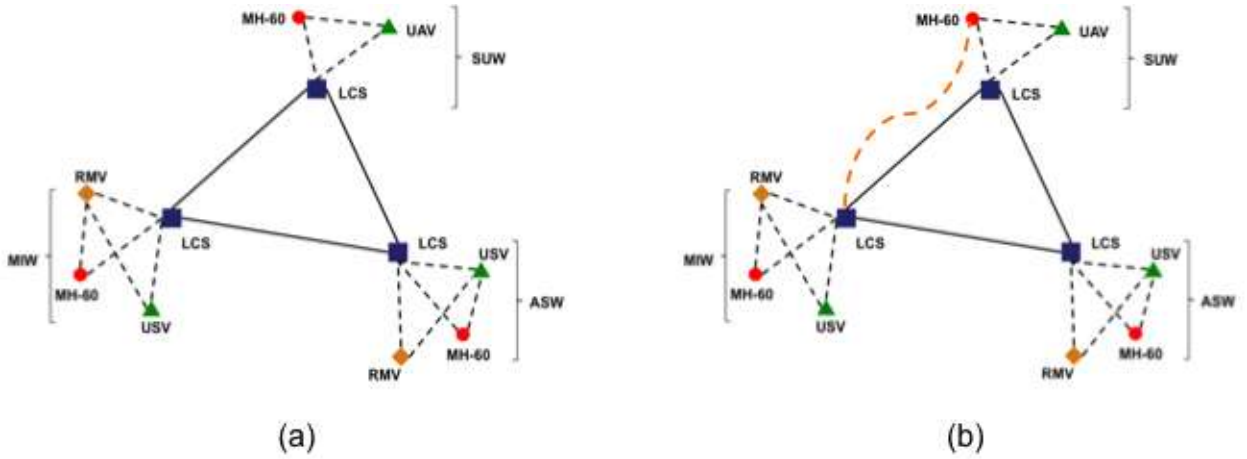

Figure 9: Two ways to realize stand-in redundancy: (a) improve system features (that is, nodes alone), and (b) improve system features and enable new communication links (that is, improve nodes and links)

For this study, three scenarios are analyzed: (1) baseline case, (2) performance degradation, and (3) re-tasked SoS. The *baseline* case indicates the original SoS in which all the systems are completely functional. This scenario establishes a performance benchmark for all the following simulations as it determines the level of performance the fully functional SoS can achieve. The *performance degradation* case investigates the impact of system failures on the performance of the overall SoS. By definition, SoSs do not fail completely when one or few of their constituent systems collapse. Instead, they experience a graceful degradation to lower performance levels. This scenario evaluates the performance loss experienced by the SoS in the event of system failures. Finally, the *re-tasked SoS* scenario captures the impact of re-tasking the remaining systems, once a unit fails. This increase in the performance level of the SoS as compared to the *performance degradation* case is the added value of stand-in redundancy.

Initial results for the first two scenarios were generated. Interesting observations included the identification of redundant systems in some mission packages, and the potential benefits of functional redundancy realized through the use of improved features on the systems. Some of the challenges encountered during these simulations runs included determination of system reliability and system costs, such as downtime costs and operations costs. Current efforts involve exploring these results further as well as addressing the abovementioned issues.

3.4.3 Current Research Efforts (2): Development of SoS Resilience Metric

Motivated by the results from the RT-36 research efforts, current work also involves the development of resilience metric to evaluate this attribute in SoSs. While stand-in redundancy can be one way to characterize the resilience of a system-of-systems, this research attempts to

establish a metric that will enable decision-makers to quantify and compare the resilience of different SoSs.

3.4.4 Summary and Future Work

The Naval Warfare Scenario (NWS) was used to verify the results of the stand-in redundancy method developed under earlier RT-36 efforts. Initial results pertaining to the impact of single system failure and re-tasking the constituent systems were obtained. In future work, the ABM will be used to: (1) determine systems most critical to the resilience of the overall architecture, and (2) study the impact of multi-system failures, and subsequently, the ability of stand-in redundancy to minimize this adverse effect. Concurrently, this study will also explore the development of a metric to capture the resilience of SoSs so as to inform design and operations decisions.

3.5 Functional Dependency Network Analysis (FDNA)

A Functional Dependency Network Analysis (FDNA) approach is adopted to deal with issues that relate the degree that interdependencies have on interoperability for a SoS.

3.5.1 Overview of Previous work from RT-36

The Functional Dependency Network Analysis (FDNA) technique (Garvey and Pinto, 2010) has been adapted to assess operability, reliability, and risk in operational networks, associated with SoS architectures. The SoS architecture is modeled as a directed network, where nodes represent either the component systems or the capabilities to be acquired. Links on the network represent functional dependencies between the constituent systems. Each dependency is characterized by strength and criticality. For operational networks, FDNA is used to assess the effect of topology and of possible degraded functioning of one or more systems on the operability of the network. The ultimate goal of the technique seeks to analyze the effects of the dependencies -and of their strength and criticality- on operability, and to identify valid operating strategies and architectures.

Root nodes are characterized by a level of operability (how good the system is doing, in a scale from 0 to 100), related to the performance (physical meaning of a certain level of operability). Operability of non-root nodes depends on their self-effectiveness, and on the operability of their predecessors. The strength of dependency is a measure of the fraction of the operability

level of a node due to the dependency. The criticality of dependency is evaluated as the maximum level of operability that a system can reach if the operability of the corresponding predecessor is 0. The method has been successfully tested on a simple five-node aerospace network. Two different types of analysis have been performed:

- *Deterministic analysis.* FDNA is used to analyze specific instances of the network that is specific levels of self-effectiveness for each of the constituent systems in the network. Analysis of the results gives good insight into the influence of dependencies on the operability of the SoS. Critical nodes, that most affect the operability of other nodes, are identified. Some emergent behavior is identified, especially in networks with greater complexity: for example, unexpected improvement in the operability of a node when some feeding capability is removed, or the architecture is modified. Comparison of architectures is another analysis that can be achieved through FDNA.

- *Stochastic analysis.* This kind of analysis requires Monte Carlo simulation, since multiple dependencies, and the use of both strength and criticality of dependencies, prevent a simple closed form for the probability density function. Stochastic analysis gives a more realistic view of the global behavior of the SoS as a function of the dependencies. The expected value for the operability of a system gives a measure of the resilience of such system to failures of the predecessors. The variance evaluates the sensitivity of the system to failures of the predecessors. Differently from the deterministic analysis, this evaluation is not based on the simulation of single instances (that could for example neglect some criticality of dependency), but it accounts for any possible combination of the effects of criticality of dependency, strength of dependency, and topology.

3.5.2 FDNA IN THE ANALYTIC WORKBENCH

In RT-44, FDNA will be part of a collection of metrics, tools, and techniques to analyze systems of systems. In particular, the deterministic analysis will be used to assess and analyze individual relevant instances of scenarios, based on the requests by the users. The stochastic version of FDNA will instead be used to analyze and identify features of the global behavior of architecture, as described above. As a first step towards the development of the analytic workbench, FDNA needs to be interfaced with a model of a real-world system of systems. The integration of FDNA method with the Naval Warfare SoS Agent-based Model is described in section 3.5.3.

3.5.2.1 Value of FDNA in the workbench

FDNA has been compared to the Bayesian approach, proposed in RT-36 to model the effect of dependency on the propagation of disruption, to clarify the differences between the two approaches. While the Bayesian approach is inherently probabilistic, and its goal is to model the uncertainties in the propagation of disruptions in a network, FDNA has a deterministic version, based on the composition of algebraic functions to model the effect of degraded operability on the operability of successor nodes.

Such formulation takes into account both the strength and the criticality of each dependency, and is suitable to be tailored to specific problems, and to analyze complex interactions between systems. One of the proposed approaches for the use of FDNA to analyze SoS involves both deterministic and probabilistic rates of failure, that of course cannot be achieved with Bayesian analysis.

In addition, while the Bayesian analysis involves "absolute" disruptions, with a probability of arising and being propagated to successor nodes, FDNA can cope with partial failures and partially degraded operability, that can be analyzed either as a deterministic value (single instance), or as a variable with a given distribution.

3.5.3 FDNA AND THE NAVAL WARFARE SOS AGENT-BASED MODEL

As aforementioned, the current effort about FDNA in RT-44 aims at interfacing the method with a model of a real-world SoS, in order to obtain the input required to perform FDNA analysis. A procedure has been developed to achieve this integration:

- The ABM architecture (fig. 8) must be analyzed and converted into an FDNA network:
    - The measure of merit for the desired analysis must be identified. For the Naval Warfare Scenario, it is the total time (sum) to achieve reconnaissance and to engage each enemy.
    - The variable measured by the operability of the agents must be identified. In our example, the probability of correct reconnaissance, and the probability of correct engagement are used.
    - If an agent can perform more than one action, it must be divided into subsystems.
    - Based on the previous points, nodes that will appear in the FDNA network are identified, keeping in mind that they can be systems or capabilities (the final nodes will be related to the desired analysis, i.e. enemy reconnaissance and engagement).

- Links are added to the network, based on the functions that are performed, and on the operational dependency (in our case, reconnaissance is dependent on the recon systems, while engagement is dependent on the engagement systems and on the reconnaissance).

The resulting FDNA network for the Naval Warfare SoS is showed in Figure 10.

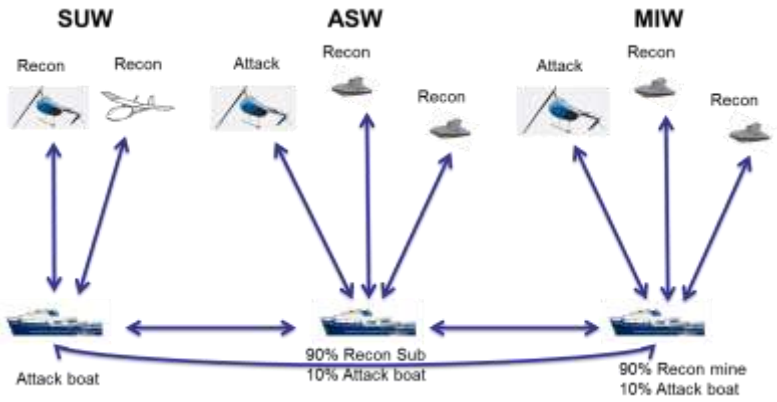

Figure 10 : One scenario of the Naval WarfareSoS. SUW is the Surface Warfare System, ASW is the Anti-Submarine Warfare System, MIW is the Anti-Mine Warfare System. Links represent communication between the agents. Performed operations are also showed.

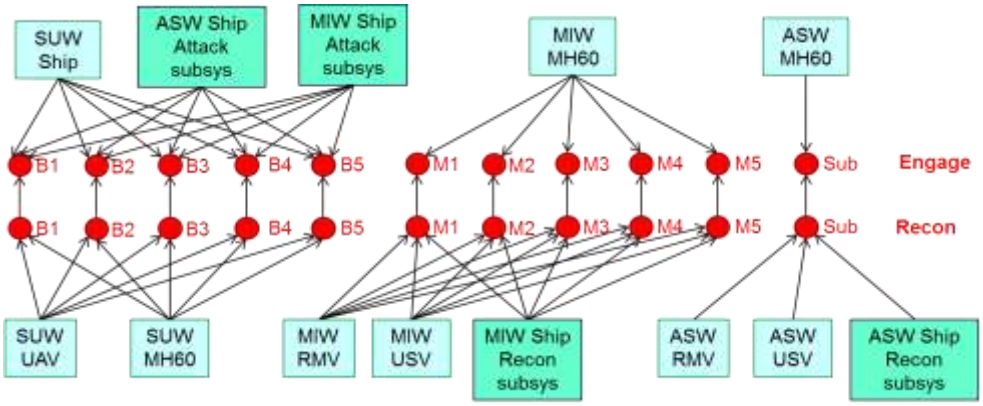

Figure 11: The resulting FDNA network. Red nodes are related to the desired analysis (B=enemy boat, M=enemy mine, Sub=enemy submarine). Blue rectangles show the agents split into subsystems.

- Further issues must be solved: when an agent is added, resulting in a link added to an already existing multiple dependency, the relationship between operability and performance must be updated, because the performance is expected to increase, even if the operability maintains the same level (Figure. 11).

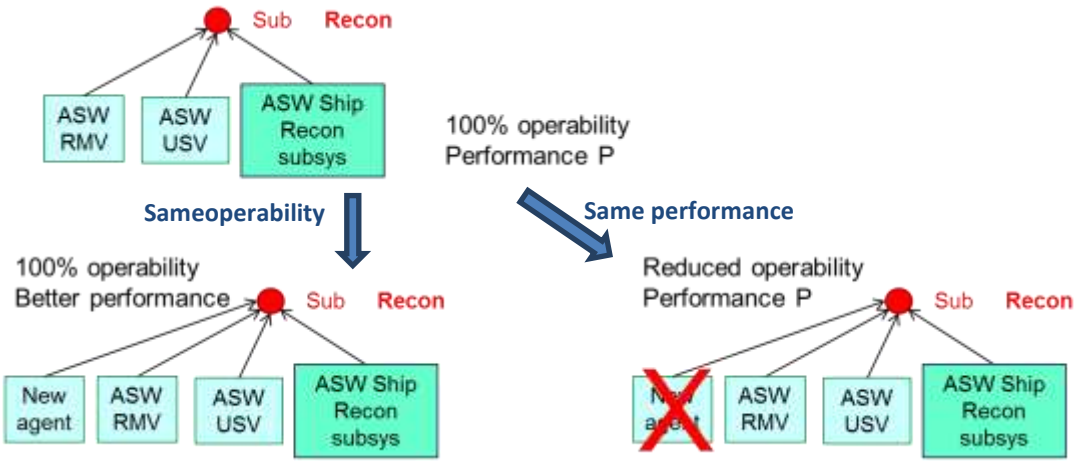

Figure 12 : Top: part of an FDNA network. When the operability is 100%, the performance is P. Lower left: when a new agent is added, a 100% operability will give a better performance. Lower right: when the new agent is not working, the operability is reduced, but the performance is P.

- Since there are no trigger events in FDNA, cases of a trigger event in the ABM must be accounted for by means of the Criticality of Dependency.
- Results from the ABM must be interpreted, in order to use them to evaluate the strength and criticality of dependencies in the network. Garvey and Pinto (2012) give insight into possible forms for the strength of dependency function, then they suggest to determine the strength (SOD) and criticality (COD) of each dependency using equations from the FDNA (that do not result in a unique solution), and then asking experts to suggest a possible scale. Instead, a Design of Experiments (DoE) will be performed to retrieve the desired value as result of a regression based on the results of multiple runs of the ABM.
- Simulations for the proposed DoE are currently underway. The goal is to use the DoE to compute performance in different instances of the scenario, SOD, and COD, and apply the retrieved parameters to difference instances of the scenario, or to different scenarios. Results from the FDNA will then be compared to results from the ABM.

3.5.4 SERC Exchange: Feedback From Dr. Garvey, MITRE Corporation

A SERC exchange with Dr. Garvey, co-creator of FDNA, facilitated by Dr. Judith Dahmann and Mr. Scott Lucero, was held in January. Dr. Garvey appreciated and approved the application of FDNA method to SoS analysis, and the interpretation given to FDNA in a different field from the original application, that is risk analysis. Different proposals for possible strategies to extend the method have been advanced for future study.

3.5.5 Conclusions and Future Work

FDNA method, successfully tested in the context of RT-36, is currently being integrated with the Naval Warfare SoS ABM. A procedure to convert the ABM network into an FDNA network has been developed, as well as methods to interpret the results from the ABM. The proposed Design of Experiment to retrieve SOD and COD from the ABM is currently underway.
Besides the integration of FDNA into the analytic workbench, future development for FDNA includes:

- Optimization of a capability portfolio through FDNA analysis.
- Parallel use of FDNA and Development Dependency Network Analysis (DDNA, section 3.6): using FDNA to assess the operability (and thus the risk and flexibility associated with a network) of a partially developed network. Since the architecture of the partially developed network is a result of DDNA analysis, the SoS authority can request a modification in the development schedule and architecture in order to increase the partial capabilities of the network of systems during development.
- Expand stochastic analysis with FDNA, testing on complex networks and performing different types of analysis. Add sensitivity analysis.

3.5.6 Publications

- *Dependency Analysis of System-of-Systems Operational and Development Networks*
  Accepted to the Conference on Systems Engineering Research, it describes the basics of FDNA and DDNA and their application to the analysis of Systems-of-Systems.

- *Maintenance and Recycling in Space: Functional Dependency Analysis of On-Orbit Servicing Satellites Team for Modular Spacecraft*
  Submitted for acceptance to AIAA Space 2013, it deals with the use of FDNA to analyze a two-level SoS composed of modular satellites, and on-orbit servicing satellites. FDNA is used to assess the operability of the SoS, and to determine if servicing is needed, and which architecture guarantees greater robustness.

- *Dependency network analysis: fostering the future of space with new tools and techniques in space Systems-of-Systems design and architecture*
  Submitted for acceptance to the International Astronautical Congress 2013, it proposes the parallel use of FDNA and DDNA to architect space SoS, with DDNA resulting in partial developed networks to be analyzed through FDNA, to assess partial capability of the SoS.

3.6 Development Dependency Network Analysis (DDNA)

The same issues about System-of-Systems engineering described in section 3.5 exist for development networks, where the links between systems represent a development dependency between managerially independent systems that comprise the SoS.

3.6.1 Overview of Previous work from RT-36

In the context of RT-36, a Development Dependency Network Analysis (DDNA) method, based on the concepts of SOD and COD from FDNA, has been developed. It is applied to development SoS networks, where the links, like in PERT, represent development dependencies between systems. The outcome of such analysis is the beginning time and the completion time of the development of each system, as well as an assessment of the combined effect of multiple dependencies and possible delays in the development of predecessors.

As in FDNA, this method evaluates the most critical nodes and dependencies, and can be used to compare different architectures in term of development time. Also, if some of the nodes are capabilities to be achieved, the method assesses the time (or the expected time, in probabilistic analysis) by which each capability is available. The method has been developed to also account for the possibility of measuring partial capabilities attained during the development of a SoS.
Compared to existing methods, such as PERT/CPM, DDNA provides more specific insight into the effects of multiple and diverse dependencies on the development of SoS. The strength of dependency affects both the beginning time and the completion time of development of a system. Differently from PERT, development of a system can start before a predecessor is complete, if the dependency has not reached criticality. Therefore, a more realistic analysis of development time is achieved. Above all, this feature allows DDNA to assess partial capabilities during the development of a SoS.

COD affects the beginning time in the same way as in PERT/CPM: a successor must wait until a critical predecessor is complete to begin development. Instead, SOD results in a less absolute dependency. The method has been tested on a simple five-node network.

3.6.2 DDNA in RT-44, Future Work, and Publications

Since the Naval Warfare Scenario is an operational network, DDNA is not involved. Current research effort in DDNA is aimed at obtaining data to test and validate the method, and at refining the method to be applied together with FDNA, as a tool to drive decisions in SoS architecting.

Different shapes for the curves relating the completion of a system and the beginning of the development of a dependent system, and the relative formulation, has been proposed and will be implemented and tested. As described in section 3.5.5, parallel use of FDNA and DDNA has been proposed, and it will be used in a paper. For future development of DDNA, cases where the component systems are independently developed must be considered. Also, literature review on advanced management and development schedule methods (beyond PERT/CPM) is required. Correlation of development through DDNA and cost, using space systems as case study, has been studied, and it will be part of a paper. Publications involving the use of DDNA are listed in section 3.5.6.

3.7 Decision Making Support on Architecture Evolution Planning

Colored Petri Nets were employed to model discrete events in general system architectures. However, the Discrete Agent Framework developed at Purdue, DAF, allows for simulation and consideration of both discrete and continuous events within the same application. We have considered the functional aspects of the CPN model to be now part of DAF's framework and are exploring multi-period decision strategies to optimally select systems, connections and policies over multi-period time windows of SoS architecture evolution.

3.7.1 Overview of Previous work from RT-36

The proposed Colored Petri Nets (CPN) simulation framework in RT-36 is regarded as an initial step (evaluation of architecture alternatives) towards exploring SoS architecture evolution. Colored Petri Nets is adopted to construct an executable architectural model to calculate performance due to its ability in modeling dynamic processes and concurrent activities. Corresponding complexity, including static complexity, functional complexity and concurrent complexity, as an indicator of evolution cost, is calculated to provide a quantitative tradeoff space with performance for decision makers. As a primary contribution, the tradeoff space aids decision makers gaining a comprehensive insight of architecture evolution. The architecture modeling process can be separated into three steps – analysis phase, synthesis phase and evaluation phase, as shown is **Error! Reference source not found.**.

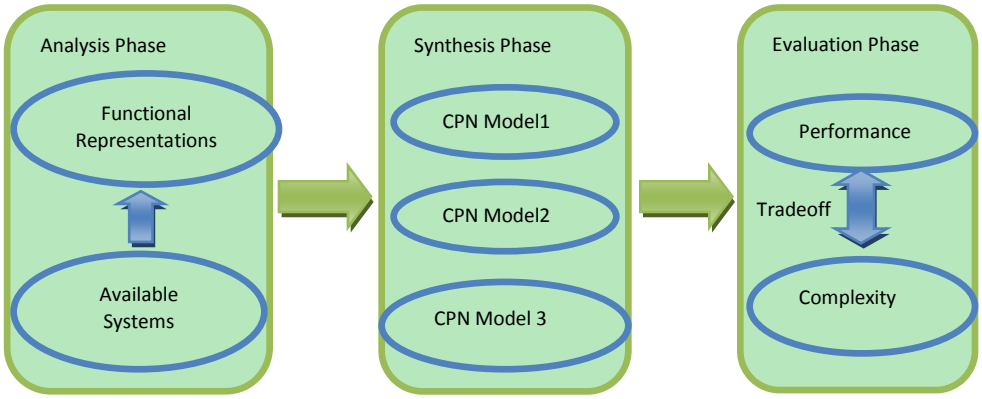

Figure 13 : Architecture Modeling Process

A simplified scenario of a Surface Warfare (SUW) module in naval warfare system of systems demonstrates the proposed approach. Results in **Error! Reference source not found.** illustrate the increased complexity along with improved performance. Arch0 only includes LCS and its NLOS missile system. Arch1 is equipped with the helicopter MH-60R and its Hellfire missile system, while Arch2 has both MH-60R and UAV with their respective Hellfire and LOGIR missile systems. Given a baseline of performance and complexity requirements, this approach would provide appropriate suggestions.

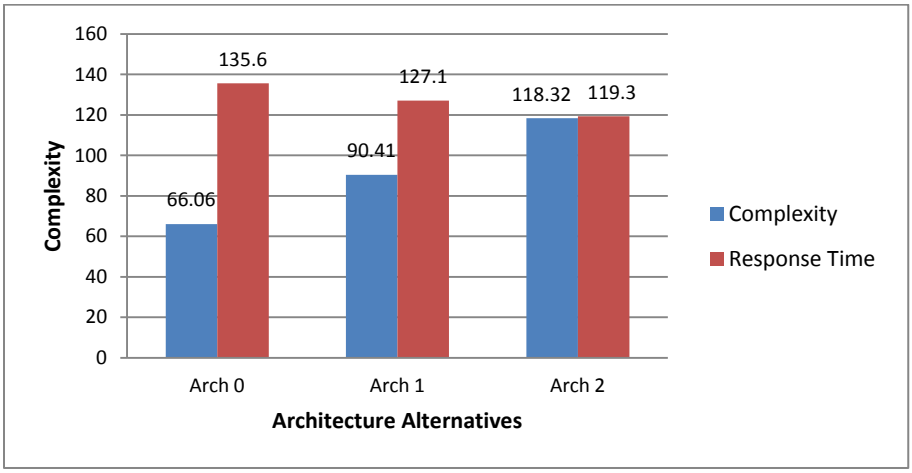

Figure 14 : Complexity versus Performance

3.7.2 Limitations of Colored Petri Nets

Concerning the ultimate goal for the research is to provide suggestions about when to apply which architecture to support architecture evolution. Towards this objective, we observe some limitations of purely using Colored Petri Nets from two perspectives. From architecture modeling perspective, each architecture needs one CPN model, which requires plenty of time to work on the modeling, especially when a large amount of architecture alternatives need to be evaluated. The heavy modeling work hinders its application to large engineering problems. From decision planning perspective, Petri net can be used to express decision tree model,

however, it is also limited to small number of alternatives and stages. Based on the two considerations, we look for a more efficient approach to conduct the analysis.

3.7.3 BASIC LITERATURE REVIEW: MULTI-PERIOD CONSIDERATIONS

We did literature review from different aspects towards SoS architecture evolution, shown in Figure 15.

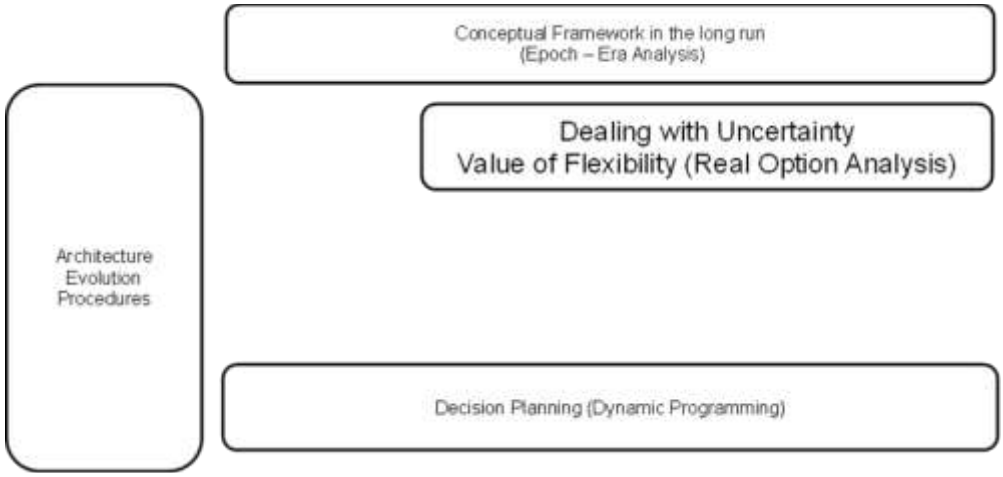

Figure 15: Overview of Literature Review

Software Architecture Evolution

Software Engineering Institute (SEI) did extensive work on software architecture evolution (Chaki, Diaz-Pace, Garlan, Gurfinkel, & Ozkaya, May 2009), (Garlan, Barnes, Schmerl, & Celiku, Sept. 2009), (Garlan & Schmerl, May 2009). They firstly distinguish between open evolution and closed-ended evolution; the former focus on management of high uncertainty, where real options are usually employed to evaluate architecture flexibility; the latter is SEI's concentration, in which current and target architectures are known and relative evolution trajectory can be generated. Moreover, SEI provides architecture evolution procedures, supported by their own tool *A*Evol, to formalize the process of architecture evolution and meanwhile categorize evolution styles in addition to architecture styles.

Epoch-Era Analysis

Epoch-Era Analysis, proposed by Ross, was used for conceptualizing system timelines using natural value-centric time scales (Ross A. R., Jun 2008). *Epoch* refers to a period with a fixed context; characterized by static constraints, available design concepts, available technology, and articulated attributes. Each *Era* is generated by stringing together sequences of epochs given likelihood of switching between given epochs and the durations of each epoch. A multiple of extended work based on epoch-era analysis were conducted at MIT, and the primary focus was the valuation of changeability under the framework (Fitzgerald & Ross, March 2012), (Fitzgerald M. R., March 2012).

Real Option Analysis

Real Option Analysis values flexibility when certain options are embedded to cope with future uncertainties (de Neufville, 2003), (Chaize, 2003), (Mikaelian, Jun 2009). De Neufville categorized real options as real options "on" projects and real options "in" projects. The former one is used to evaluate projects worthy to invest or not accounting for possible options of dropping, expanding, delaying, etc, in the future. While the latter one evaluates the value of flexibility designed in system, such as modularity, changeability and so forth. It is useful for initial architecture selection embedded future options; however, constraints are also obvious. One is that those valuation techniques from financial options can only be applied when engineering systems can be fit into financial option model.

System Decision Planning

Previous literature review primarily lay outs framework for architecture evolution analysis. For actual analysis and calculation, two intuitive techniques are investigated in system decision planning, decision tree analysis and dynamic programming, particularly, time-expended decision networks and approximate dynamic programming.

- Time Expanded Decision Networks

Silver (Silver & de Weck, 2007) developed concepts of Time-Expanded Decision Networks (TDNs) for designing and analyzing flexibility in large-scale complex systems. It synthesizes concepts from decision theory, real option analysis, network optimization and scenario planning and aims to generate "best" development and operational paths for decision makers. It essentially uses shortest-path algorithm to calculate paths with minimum costs. Computation cost is smaller than traditional decision tree analysis however it is suitable for analyzing a limited number of configurations for a few major time periods.

- Approximate Dynamic Programming

Approximate Dynamic Programming (ADP) was developed by Powell and extensive applications using ADP were completed (Powell, 2010). Classical dynamic programming, typically uses backward induction approach, usually suffers from curses of dimensionality – the explosion of state space, the outcome space and the action space. Approximate dynamic programming is intended to solve these difficulties. The foundation of ADP is based on an algorithm strategy that steps forward through time (also named as "forward dynamic programming"), that is, value function needs to be approximated iteratively to converge to an optimal result. ADP has been applied to a number of real large-scale engineering problems, such as solving resource allocation problems under uncertainty in transportation and logistics field, or optimizing the design and control of storage portfolios in energy field. Based on these concerns, ADP is selected as our candidate approach to deal with SoS architecture evolution, and more details will be shown below.

3.7.4 APPROXIMATE DYNAMIC PROGRAMMING (ADP)

The five fundamental elements in dynamic programming are states, actions, exogenous information, transition function and the objective function. Assume system is in a state S_t, from which an action x_t is taken and then new information W_{t+1} is observed, which takes system to a new state S_{t+1} by a transition function $S^M(\square)$. We can represent the rule or policy for making a decision using the function $X^\pi(S_t)$. There is a choice of decision functions X^π where $\pi \in \Pi$ designates a particular function or policy. The dynamics of the problem can be expressed using

$$x_t = X^\pi(S_t) \tag{14}$$

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}) \tag{15}$$

It is also assumed that contribution (cost) can be computed given $C(S_t, x_t)$ depending on state S_t and decision x_t (sometime even W_{t+1}). Then the objective function is to find the best policy $\pi \in \Pi$ that solves

$$\min_\pi E^\pi \left\{ \sum_{t=0}^{T} \gamma^t C\left(S_t, x_t\right) \right\} \tag{16}$$

where γ is a discount factor. And expectation is necessary because the information variable W_t is random.

This optimization problem can be solved through recursively computing the following optimality equations backward through time for very small problems.

$$V_t(S_t) = \min \ (C_t(S_t, x_t) + \gamma E\{V_{t+1}(S_{t+1}) \mid S_t\}) \tag{17}$$

where $V_{t+1}(S_{t+1})$ is the value function which gives the expected value of being in state S_{t+1} at time $t+1$ and following an optimal forward policy.

However, when our state variables, decision variables and exogenous information have multiple dimensions, it cannot be solved by backward induction. With approximate dynamic programming, we step forward in time. Then two problems need to be solved: one is to find a way to make decisions; the other is to find a way to randomly generate a sample of what might happen. Since we step forward in time, we have not computed exact value function, thus we need to find an appropriate approximation of value function. And because future information is unknown to us, a sequence of sample realizations of random exogenous information needs to be generated, which is usually obtained by three ways: real world process, computer simulations, and sampling from a known distribution. Moreover, multiple approaches can be employed for better approximation, for instance, approximate value iteration can eliminate the assumption that we can compute the one-step transition matrix; the post-decision state variable can be used to simplify the process of approximating the expectation.

Overall, ADP is a promising framework to support sequential decision problems. The general framework, when applied to the SoS evolution context, translates to optimally selecting the

policies for individual systems and the way they are connected in achieving some evolving SoS capability.

3.7.5 EARLY FORMULATION

A synthetic example is formulated using and ADP approach. Assume there are nine systems $\{S_1, S_2, ..., S_9\}$, among which some are already available, some are in the process and some are unavailable currently. This engineer has to make decisions about which systems should be put together to provide SoS capability at which stage. We assume three stages $\{T_1, T_2, T_3\}$ are investigated, all other assumptions are shown in Table 3.

Table 3 Assumptions Illustration

Elements	Expression
Decision variables	$[x1, x2, \ldots, x9]_{Ti}$ (xi can be 0 or 1)
State variables	Full Capability: $(Cp1, Cp2, \ldots, Cp9)_{Ti}$; Capability Requirement: Rq_{Ti} Cost: $(Cs1, Cs2, \ldots, Cs9)_{Ti}$; Budget: B_{Ti} Risk: $(Rs1, Rs2, \ldots, Rs9)_{Ti}$; Risk Acceptance: Ra_{Ti}
Exogenous information	Technology Readiness Level (TRL)
Transition function	$Cs_{T(i+1)} = Fc\ (Cs_{Ti}, TRL_{T(i+1)}, X_{Ti})$ [assume: $Fc = Cs_{Ti} - \alpha * X_{Ti} - \beta * TRL_{T(i+1)}$] $Rs_{T(i+1)} = Fr\ (Rs_{Ti}, TRL_{T(i+1)}, X_{Ti})$ [assume: $Fr = Rs_{Ti} - \mu * X_{Ti} - \nu * TRL_{T(i+1)}$]
Objective function	Max: $F = Cp_{T1} * X_{T1} + Cp_{T2} * X_{T2} + Cp_{T3} * X_{T3}$
Functions	F1: S1, S2, S3; F2: S4, S5, S6; F3: S7, S8, S9
Integration Rules	Cp = sum(Cps), C = sum(Cs), R = sum(Rs); At least one system for a function should join; S1 and S6 are not compatible
Evolution Rules	Transition function; S1 and S6 are not compatible

3.7.5 Conclusion and Future Work

The literature review has completed and approximate dynamic programming is chosen as a primary approach for next step towards SoS architecture evolution. An example has been formulated. So the near term work is to construct and implement a NWS inspired example. Next step would be investigation of those assumptions and inputs involving where and how to obtain the data; meanwhile the application to the Naval Warfare Scenario SoS will be explored.

3.8 Robust Portfolio Optimization

Current guidelines and tools for architecting SoS architectures are lacking sufficient capabilities in enabling effective decision-making for SoS SE practitioners. This section builds upon advances in the development of decision analysis framework that treats SoS architectural evolution within an investment portfolio framework. Work in this section extends initial frameworks developed under the Naval Postgraduate School (NPS) Acquisition Research Program, and subsequently through SERC sponsored RT-36, towards SoS architectural management. Methods and mathematical frameworks from robust optimization and financial engineering practices are adapted to serve the purpose of evolving SoS architectures.

3.8.1 Prior Work: A Brief Overview

We adapt frameworks from operations research and financial portfolio investment strategies in addressing SoS architectural management. In the context of an SoS, the expected returns correspond to a desired capability from an investment in a system, and the variance (covariance) can be attributed to developmental or operational risks that arise both from the inherent dynamic of the chosen system, and its interactions with connected systems. In this research, a generic, operational SoS architecture is modeled as an interconnected set of discrete nodes; each having a finite set of inputs and outputs. The interconnectivities between nodes are established to facilitate the fulfillment of individual node requirements by allowing for node capabilities (outputs) from existing nodes to connect and consequently fulfill requirements (inputs) of any compatible node requiring a particular capability to function. Overarching capabilities are provided by nodes that directly contribute to these required capabilities.

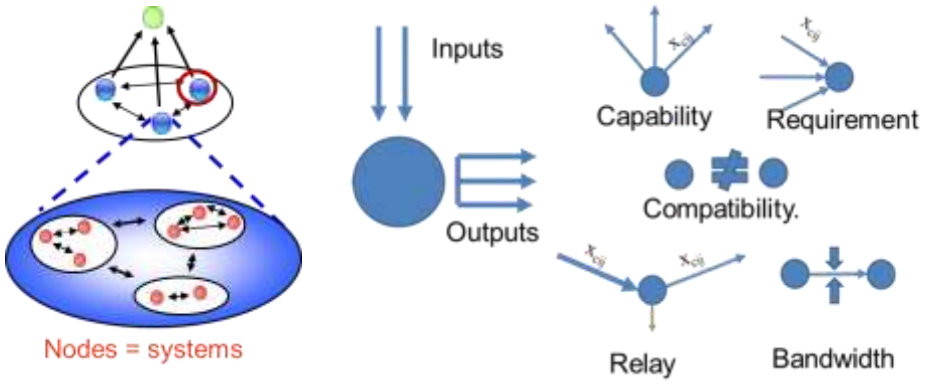

Figure 16: (a) SoS hierarchy (b) nodal behaviors

Figure (16a & b) shows the five archetypal interactions of:

- *Capability:* Nodes have finite supply of capabilities that are limited by quantity and number of connections.
- *Requirements:* Nodes have requirements to enable inherent capabilities. Requirements are fulfilled by receiving connections from other nodes that possess a capability to fulfill said requirements.
- *Relay:* Nodes can have the ability to relay capabilities between adjacent nodes. This can include excess input of capabilities that are used to fulfill node requirements.
- *Bandwidth:* Total amount of capabilities and number of connections between nodes are bounded by 'bandwidth' of the connection linkages between systems.
- *Compatibility:* Nodes can only connect to other nodes based on a pre–established set of connection rules.

The performance of the SoS is related to the ability of the connected network of individual systems to fulfill overarching core objectives. The SoS-wide performance is quantified by the capability of nodes that most directly contribute to the core objectives. It is assumed that these core objectives can be at least, approximated quantitatively.

Robust Portfolio Optimization for SoS

The SoS investment model is posed as a robust mean variance optimization problem. The objective here is to maximize the expected network performance in fulfilling key overarching objectives whilst minimizing acceptable levels of developmental risk and cost. Selection of system is constrained by fulfilling generic connectivity requirements between constituent systems. The robust mean variance formulation uses a semi-definite programming (SDP) approach to deal with uncertainties in the estimated performance of individual systems, and

their development risks. Additionally, a measure of robustness for linear constraints was implemented, using a Bertsimas-Sim formulation, to address control of operational robustness that provides probabilistic guarantees on potential constraint violations.

The robust mean-variance portfolio (RMVO) approach was demonstrated for a Littoral Combat Ship (LCS) inspired concept scenario. The goal was to generate a tradeoff efficiency frontier between SoS capability and potential development risk, where discrete points on the frontier represent 'optimal collections' of interconnected systems, at accepted levels of SoS level performance vs. developmental risk. Additionally, the Bertsimas-Sim approach was adopted to enforce probabilistic guarantees on the communications layer for a concept LCS problem as well. Results of the frontiers for each method (RMVO and Bertsimas-Sim) as shown in Figure (17).

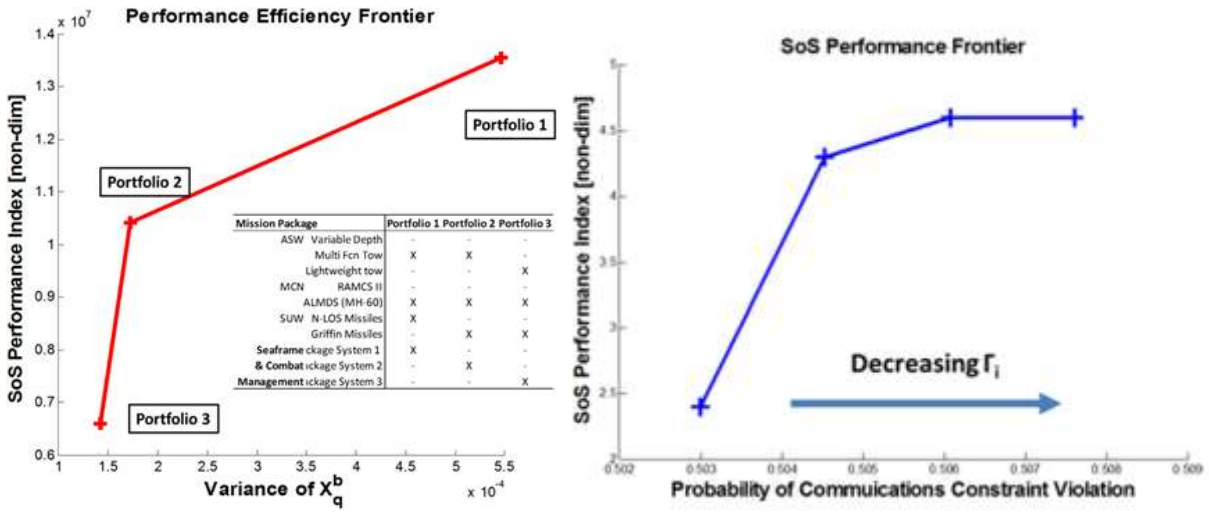

Figure 17: (a) SoS efficiency frontier (b) Operational constraint violation frontier

Detailed results and discussion on the mathematical formulation and solution to the concept example for Figure (4) above are as published in the SERC RT-36 Final Report (Report Number: SERC-2012-TR-036).

3.8.2 CURRENT PROGRESS IN ROBUST PORTFOLIO OPTIMIZATION APPROACH

The following is a list of capabilities and modifications that have been made to the portfolio formulation to facilitate a more detailed representation of system attributes and functions that are considered in the optimization process.

3.8.2.1 Modeling diminishing returns ("Economies of Scale")

Typical trends in acquisitions reflect diminishing returns (gains from capability) with incremental addition of a particular capability. The objective function of the robust portfolio optimization problem, as developed in work under RT-36 considers a purely linear utility of gain (discrete or continuous). The prior optimization framework is extended to include improvements to better capture 'value-added' characteristics through the introduction of constrained, *piece-wise linear* approximations to capture the non-linearity of diminishing returns (economies of scale) for systems that exhibit such behavior. Piece-wise constraints, while not strictly linear in their native form, can be converted to a linear program (Magnani, 2009). The main challenge is in the choice of piece-wise linear approximations used to represent the original nonlinear conditions. The concavity of diminishing returns, makes its piece-wise linear approximation generally easy to solve.

3.8.2.2 Robust Linear Constraints (Added Conic Approach)

The Bertsimas-Sim approach, as introduced in earlier work for providing probabilistic guarantees on constraint violations for operational constraints, deals with uncertainties in the [A] matrix for a general set of inequality constraints [A]{x}={b}. We extend our consideration of uncertainty in linear operational constraints to account for uncertainties in both the entries of [A] and {b}. This is accomplished through first parameterizing the uncertainties in [A] and {b} within the following *elliptical uncertainty* set:

$$\mathrm{U} = \left\{ \left[\mathbf{A^0; b^0} \right] + \sum_{j=1}^{k} u_j \left[\mathbf{A^j; b^j} \right], \|u\| \le \mathbf{1} \right\} \tag{18}$$

The conic optimization approach developed in literature (Nemirovski, 1998) assumes the above parameterization for uncertain linear coefficients and results in the following robust constraint:

$$(\mathrm{A}^0)^{\mathrm{T}}\mathrm{x} + \mathrm{b}^0 - \sqrt{\sum_{j=1}^{k} ((\mathrm{A}^j)^{\mathrm{T}} x + \mathrm{b}^j)^2} \ge 0 \tag{19}$$

It must be noted that the uncertainties here are assumed to be *constraint-wise* uncertainties. The robust inequality in Eq. (4) can equivalently be rewritten into the following form:

$$z_j = (A_j)^T x + b^j, j = 0, \ldots, k \quad (20)$$

$$(z_0, z_1 \ldots\ldots, z_k) \in C_q \quad (21)$$

where C_q is a second order cone. Although the resulting problem is nonlinear, it is nevertheless convex, amenable to highly efficient interior point algorithms and solvable in polynomial time for continuous variables. In the case of integer variables, the problem is NP-hard due to the integrality condition. The combination of state-of-the-art integer solvers with the underlying interior-point algorithm to solve SOCPs makes the problem quite tractable for reasonably sized dimensions. The inclusion of the conic form of robust linear constraint, as shown in the equations above allows for strict feasibility constraints to be set, assuming uncertainty in both [A] and {b}; this can reflect, say, the uncertainties in power generated by systems [A] to meet an uncertain demand condition {b}.

3.8.2.3 Alternative Robust Implementation

The semidefinite programming (SDP) approach to addressing uncertainties, as a robust portfolio optimization problem, presents challenges in implementation for an analytic workbench. This is particularly true for the formulation of the portfolio problem from RT-36 that is formulation as a Mixed Integer Semi Definite Program (MISDP), to which only 1 academic solver is available for its solution (at the time of this report).

The consideration of the workbench to be openly shareable and deployable prompts the need for alternative parameterizations of robust formulation as a mathematical program of less complexity, but comparable performance. The conversion may translate to the parameterization of risk (covariance of development time in the MISDP formulation) to another correlated form that does not present quadratic uncertainty in risk. Promising linear and alternative conic approaches are being explored ((Fabozzi, 2007), (Powell W. R., 2012)).

3.8.3 Future Work

Additional measures have been introduced to the modeling aspects in the robust portfolio framework. The measures include the consideration for economies of scale, conic approach to uncertainties and continued work on the multi-period nature of sequential portfolios. Future steps towards Phase II will include a more comprehensive application based on architectural decisions for the NWS model. The methods will be combined with aspects of ADP policies to facilitate sequential portfolio selections in an SoS evolutionary context.

4 SUMMARY AND FUTURE WORK

This report has detailed both new results achieved during Phase 2 under RT-44b and extensions of previous results from Phase 1, towards the development of an analytic workbench that uses a suite of MPTs in support of evolving a SoS architecture. We have refined the methods further in enabling more detailed analysis of alternatives in a SoS environment, through instructive example of application on a concept Naval Warfare Scenario (NWS) case study. The NWS has also been further developed towards a higher-fidelity of capability, to show the benefits of agent-based simulations and to serve as a synthetic platform for application of our analytic workbench methods.

Our further work under RT-44 Phase II will continue efforts towards refining and maturing our analytic workbench; this includes more instructive applications of each method on the NWS agent simulation, as appropriate. The work will also:

- Implement more realistic scenarios/assets for the concept NWS problem, to facilitate more instructive demonstration of the analytic workbench.
- Refine metrics for the NWS problem in a SoS context and determine which metrics can be computed, and under which assumptions, for each candidate analysis method.
- Demonstrate, within the structure of an analytical workbench, the application of researched methods in evolving the NWS model. This involves application of each method to representative architectural challenges that SoS practitioners may face in developing the NWS problem.
- Validation and verification of efficacy of methods in the context of the extended NWS problem; this can be accomplished through feedback from sponsors and various participating groups that have agreed to confer with us, through our sponsors. This includes entities in the US Navy and MITRE Corporation respectively.
- We have future outreach efforts through an INCOSE webinar and an NDIA webinar that are scheduled for later parts of this year. Sponsors have indicated the importance of these webinar sessions that are to showcase research in our RT 36 & 44 efforts, in both facilitating feedback from industry and military experts in systems architectures. These sessions also disseminate the existence of the analytic workbench as a potential beta-tool for testing in the future.
- Open source considerations are explored. At the moment, our MATLAB environment software facilitates earl platform development to facilitate sharing; this includes a potential GUI environment for beta versions of the platform.

Appendix: References

Abbot, B.P. 2008. "Littoral Combat Ship (LCS) Mission Packages: Determining the Best Mix." Master's thesis. NavalPostgraduate School.

Blanchard, B., and Fabrycky, W., *System Engineering and Analysis,* 3rd ed., Prentice Hall International Series in Industrial and Systems Engineering, 1998, Chaps. 1, 2, Appendix A.

BertsimasSim,M.,D.,. (2004). The Price of Robustness. "Operations Research", 52(1), 35-53.

Blanchard, B., & Fabrycky, W. (1998). *System Engineering and Analysis, 3rd ed.* Prentice Hall International Series in Industrial and Systems Engineering.

Bocchetti, D., Giorgio, M., Guida, M., & Pulcini, G. (2009). A competing risk model for the reliability of cylinder liners in marine Diesel engines. *Reliability Engineering and System Safety, 94*, 1299-1307.

Briand, D., Lowder, K., & Shirah, D. (October 2006). *Updating Time-To-Failure Distributions Based On Field Observations and Sensor Data.* Albuquerque, New Mexico: Sandia National Laboratories.

Chaize, M. (2003). Enhancing the Economics of Satellite Constellations via Staged Deployment and Orbital Reconfiguration. MIT.

Chaki, S., Diaz-Pace, A., Garlan, D., Gurfinkel, A., & Ozkaya, I. (May 2009). Towards Engineered Architecture Evolution. *Modeling in Software Engineering*, (pp. 1-6).

de Neufville, R. (2003). Real Options: Dealing with Uncertainty in Systems Planning and Design. *Integrated Assessment, Vol.4*(No.1), 26-34.

Dhillon, B. (2006). *Maintainability, Maintenance, and Reliability for Engineers.* CRC Press.

Fitzgerald, M., & Ross, A. (March 2012). Mitigating Contextual Uncertainties with Valuable Changeability Analysis in the Multi-Epoch Domain. *Systems Conference (SysCon), 2012 IEEE International*, (pp. 1-8).

Fitzgerald, M., & Ross, A. (March 2012). Sustaining Lifecycle Value: Valuable Changeability Analysis with Era Simulation. *Systems Conference (SymCon), 2012 IEEE International*, (pp. 1-7).

Garlan, D., & Schmerl, B. (May 2009). AEvol: A Tool for Defining and Planning Architecture Evolution. *International Conference on Software Engineering*, (pp. 591-594).

Garlan, D., Barnes, J., Schmerl, B., & Celiku, O. (Sept. 2009). Evolution Styles: Foundations and Tool Support for Software Archtiecture Evolution. *Software Architecture, 2009 & European Conference on Software Architecture. WICSA / ECSA 2009*, (pp. 131-140).

Magnani Boyd, S.A.,. (2009). Convex piecewise-linear fitting. "Optimization and Engineering", 10, 1-17.

Mikaelian, T. (Jun 2009). An Integrated Real Options Framework for Model-based Identification and Valuation of Options under Uncertainty. MIT.

Nemirovski Ben-Tall and A.A. (1998). Robust Convex Optimization. "Mathematics of Operations Research", 769-805.

Powell, W. (2010). *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (2 ed.).

Ross, A., & Donna, R. (Jun 2008). Using Natural Value-Centric Time Scales for Conceptualizing System Timelines through Epoch-Era Analysis.

Silver, M., & de Weck, O. (2007). Time-Expanded Decision Networks: A Framework for Designing Evolvable Complex Systems. *Systems Engineering, Vol.10*(No.2), 167-186.